**Муниципальное бюджетное образовательное учреждение города Костромы**

**«Лицей № 17»**

Проект по информатике

«Голосовой ассистент на языке Python»

Работу выполнил:

Сахаров Роман Максимович,

учащийся 10 «А» класса

Руководитель:

Виноградова Юлия Николаевна,

учитель информатики

**Кострома, 2019**

Оглавление

**Введение**

**Основная часть**

1. Языки программирования
2. Разработка голосового помощника
3. Общий механизм работы
4. Требования к функционалу
5. Организации логики бота
6. Модули
7. Структура программного кода
8. Работоспособность итоговой программы

**Заключение**

**Список использованной литературы**

Введение

В последние время в нашу жизнь всё больше проникают компьютерные технологии. Они используются повсеместно и играют важную роль во множестве сфер человеческой деятельности. В частности, всё большую популярность приобретают голосовые помощники.

*Виртуальный ассистент[]* – это мобильный программный агент, который может выполнять задачи (или сервисы) для пользователя на основе информации, введённой пользователем, данных о его местонахождении, а также информации, полученной из различных интернет-ресурсов. Примерами такого рода агентов являются Siri, Google Assistant, Алиса и другие.

Однако, такие ассистенты в основном выполняют общий спектр задач. Их функционал заточен под среднего пользователя и его нужды. Но разные группы людей могут нуждаться в различных функциях такой программы, а, следовательно, и в определённых её особенностях. Например, возможности и логика бота-консультанта интернет-магазина могут сильно отличаться от тех же особенностей бота, который генерирует анекдоты. Один помощник не сможет вместить в себя все возможные функции для пользователей и различных их вариаций. Для помощи определённым группам людей создаются отдельные специализированные программы и ассистенты.

Но почему бы не сделать специального бота для школьника, помощника, который обладал бы функционалом, заточенным под нужды учеников? Я не смог найти примеров реализации подобной идеи.

**Проблема:** в настоящий момент отсутствует пример программы голосового помощника для школьника, хотя идея создания таковой мне кажется удачной и интересной, а сам такой ассистент может быть потенциально полезен достаточно большой группе людей.

**Актуальность:** сейчас набирает популярность концепт так называемых голосовых помощников. Также в систему образования активно внедряются IT-технологии: электронный журнал, электронные учебники, Интернет, как источник дополнительной информации. Пример же реализации продукта, совмещающего в себе обе эти тенденции, может оказаться весьма успешным и полезным.

Таким образом, я **поставил следующую цель:** разработать собственного голосового ассистента, функционал которого будет направлен на помощь школьнику в образовательном процессе.

**Для выполнения поставленной цели я должен выполнить следующие задачи:**

1. Изучить информацию о программировании ботов
2. Предъявить требования к продукту проекта (программе-голосовому помощнику)
3. Выбрать подходящие модули и изучить документацию по ним
4. Спроектировать программу (классы, модули, функции, методы)
5. Написать механизм логики бота
6. Написать все исполнительные функции
7. Создать удобный голосовой и графический интерфейсы программы
8. Протестировать и отладить программу
9. Сравнить получившегося голосового помощника с уже существующими
10. Сделать вывод о преимуществах и недостатках получившегося продукта, о возможных перспективах развития проекта

**Проектный продукт:** программа, представляющая из себя голосового ассистента.

**Методы работы:**

1. Изучение литературы по ЯП Python 3, SQL, HTML
2. Поиск информации в Интернете
3. Программное проектирование
4. Программирование на ЯП Python 3
5. Тестирование и отладка программы

Основная часть проекта

**I. Языки программирования**

Большинство современных голосовых помощников написаны на C++ (например Алиса, Google Assistant, Microsoft Cortana). Объясняется это в целом большой популярностью языка, а также очень быстрой скоростью работы программ на нём. Однако, всё большую популярность, в том числе и для создания ИИ (искусственного интеллекта), набирает *Python*.

**Python[]**— высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций.

Python поддерживает структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное программирование. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений, высокоуровневые структуры данных. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты.

**Основные преимущества Python:**

1. *Низкий порог вхождения:* человеку, знакомому с программированием, достаточно небольшого количества времени, чтобы начать писать на нем полезные для себя скрипты, а не знакомому – Python позволяет легко открыть для себя программирование и попробовать свои силы в нем.
2. *Хорошо спроектирован:* Python вобрал в себя современные тенденции в программировании «с нуля». Кроме того, он динамично развивается: процесс включения новых конструкций в язык хорошо отлажен, и он продолжает впитывать в себя приемы функционального программирования, аспектно-ориентированного программирования и прочего, оставаясь при этом обратно-совместимым и внутренне непротиворечивым.
3. *Легко читаемый синтаксис* (по сравнению с С++, Perl, РНР): позволяет легко читать чужой код, разбираться в давно написанном собственном коде. В сочетании со сказанным выше это настраивает создателей библиотек на простоту и логичность интерфейсов.
4. *Огромное количество библиотек* с кодом на любой случай жизни: будь то работа с таблицами Excel, изображениями, сетью Twitter или даже написание нейросетей.
5. *Переносимость:* Python реализован под всеми распространенными операционными системами и на множестве архитектур – Windows, Linux, MacOS, даже на мини-компьютерах Arduino. Система зависимостей хорошо продумана, и разворачивание приложений на другой машине происходит легко и без сюрпризов.

**Почему мой выбор – именно Python?**

Во-первых, я сам хорошо знаком с этим ЯП. Я изучаю его уже почти 2 года, и на данный момент он является самым понятным, простым и хорошо изученным. Чтение литературы по Python, а также занятия в Яндекс Лицее помогли мне определиться с выбором ЯП и в конечном счёте разработать программу-продукт.

Во-вторых, процесс написания кода на этом языке довольно быстрый. Этому способствует большое количество библиотек и литературы по ним.

**“Вспомогательные” языки.**

В процессе разработки, кроме ЯП Python были использованы также *SQL* и *HTML*.

**SQL[]** (Structured Query Language — Структурированный язык запросов) — язык управления базами данных для реляционных баз данных. Сам по себе SQL не является Тьюринг-полным языком программирования, но его стандарт позволяет создавать для него процедурные расширения, которые расширяют его функциональность до полноценного языка программирования.

**HTML[]** (HyperText Markup Language — «язык гипертекстовой разметки») — самый базовый строительный блок Web. Он определяет содержание и структуру веб-контента. Другие технологии, помимо HTML, обычно используются для описания внешнего вида/представления (CSS) или функциональности/поведения (JavaScript) веб-страницы.

Одной из особенностей моей программы является возможность добавления нескольких пользователей со своими настройками. Удобнее всего хранить пользовательские данные в SQL БД (базах данных). Также для более удобного вывода информации большая её часть выводится в окно браузера графического интерфейса программы. А любая информация, выводимая таким образом должна быть предварительно преобразована в HTML-код.

Таким образом для разработки голосового помощника я использовал Python 3, SQL, HTML

II. Разработка голосового помощника

**Общий механизм работы**

***Речевая команда***

***Ответ текстом + звуком***

***Программа***

**При очередной пользовательской реплике:**

1. Пользователь произносит в микрофон команду
2. Программа принимает её и переводит в текст
3. Команда (в текстовом виде) обрабатывается
4. Определяется, что нужно сделать в соответствии с запросом пользователя (определяется вид команды)
5. Выполняется соответствующее действие
6. Формируется ответная реплика бота
7. Слова бота озвучиваются и выводятся на экран

**Требования к функционалу**

В связи с тем, что данный голосовой ассистент разрабатывается для школьника, мной были выдвинуты следующие требования к элементам функционала программы. Ассистент должен:

1. Обращаться к пользователю по имени
2. Приветствовать пользователя при входе в систему
3. Быть вежливым с пользователем
4. Уметь сказать, сколько времени
5. Уметь открывать электронные книги и учебники
6. Уметь говорить расписание/домашнее задание
7. Искать информацию по запросу в интернет-ресурсах
8. Уметь пошутить, если пользователь попросит
9. Прощаться перед закрытием программы

На мой взгляд, этот список функций голосового помощника является достаточным для определённой помощи школьнику в образовательном процессе.

**Организация логики бота**

Большинство современных голосовых ассистентов используют нейросети и машинное обучение в организации логики. Однако, как написание такого ИИ (искусственного интеллекта), так и его обучение – это очень долгий, трудоёмкий и ресурсозатратный процесс. К тому же для лучшего обучения ИИ нужно большое количество людей, которые будут “общаться” с ботом, тестировать программу.

В связи с этим мной было принято решение использовать другой механизм реализации логики бота. Упрощённо суть данного способа можно описать так: в пользовательской реплике ищутся ключевые слова, на основании которых, программа “понимает”, что от неё требуется.

Таким образом, каждому элементу функционала поставлен в соответствие набор определённых ключевых слов, по которым программа и распознаёт вид команды. При этом в реплике пользователя не обязательно должно встречаться абсолютное совпадение с одним из таких слов. Вычисляется совпадение слов в процентах. Далее, находится ключевое слово с наибольшим совпадением. Если процент совпадения не меньше определённого, то программа определяет свои действия в соответствии с этим ключевым словом.

**Модули**

Для наиболее успешной реализации задуманного концепта я выбрал следующие модули Python (библиотеки):

1. Sys - обеспечивает доступ к некоторым переменным и функциям, взаимодействующим с интерпретатором python.
2. Os - модуль предоставляет множество функций для работы с ОС
3. Speech Recognition – модуль для распознавания речи
4. Fuzzy Wuzzy – библиотека для нечёткого сравнения
5. Pyttsx-3 – модуль для синтеза речи
6. SQLite3 – библиотека для работы с БД
7. WinSound – библиотека для воспроизведения звуковых файлов
8. Random – генератор случайных чисел, последовательностей, списков
9. PyQt5 – включает в себя множество библиотек для создания GUI-приложений
10. Time, datetime, deltatime – модули для работы со временем
11. WikipediaAPI – библиотека для API доступа к материалам с Wikipedia.org
12. Selenium – модуль, используемый для создания web-браузера под управлением программы

**Структура программного кода**

Скриншоты кода: *см. приложения**1-2*

Вся программа разделена на два отдельных модуля: основной (main) и дополнительный функциональный (my\_web). Второй модуль включает в себя 5 функций для работы с Интернет-ресурсами: Яндекс поиском, Википедией и электронным дневником.

* Функция «gen\_link» генерирует ссылку на запрос в Яндексе на основании поступившего в неё запроса.
* Функция «search\_in\_wiki» производит поиск в свободной энциклопедии Википедия и возвращает результат, состоящий из первых двух предложений статьи.
* Функция «netschool» производит парсинг расписания и домашнего задания из электронного дневника, используя передаваемые в неё логин, пароль и дату нужного дня
* Функции «homework» и «logout» являются вспомогательными для работы предыдущей функции.

Основной модуль состоит из двух вспомогательных функций и основного класса «Bot», наследуемого от QMainWindow, т.е. класса главного окна графического интерфейса программы. В этом классе, а точнее в его методах как раз и прописан весь основной функционал ассистента.

**Работоспособность итоговой программы**

В процесс создания практически любой программы входит *отладка[]*, т.е. процесс поиска и устранения ошибок в программе. После отладки программа является рабочей (по результатам проведённых мной тестов), однако на случай непредусмотренных ошибок и неисправностей в программу встроен механизм, который запишет все возникшие ошибки в файл логов, после чего пользователь сможет связаться с разработчиком и предоставить этот файл для дальнейшего устранения неисправности и модификации программы.

Первоначальная версия программы подразумевала, что бот будет реагировать на слова пользователя, только когда обращаются непосредственно к нему, т.е. в пользовательской реплике должно звучать заданное боту имя (одной из особенностей моего помощника является возможность задать ему любое имя в настройках). Однако, в ходе тестирования выяснилось, что это довольно неудобно, и было принято решение изменить программу таким образом, чтобы помощник реагировал на все реплики пользователя. Если пользователь не хочет обращаться к боту, он может просто выйти из программы и потом при необходимости вновь запустить её.

Скриншот с примером диалога с ботом: *см. приложения 3*

Заключение